

# MIXTURES OF SPARSE AUTOREGRESSIVE NETWORKS

**Marc Goessling**

Department of Statistics

University of Chicago

Chicago, IL 60637, USA

goessling@galton.uchicago.edu

**Yali Amit**

Departments of Statistics and Computer Science

University of Chicago

Chicago, IL 60637, USA

amit@galton.uchicago.edu

## ABSTRACT

We consider high-dimensional distribution estimation through autoregressive networks. By combining the concepts of sparsity, mixtures and parameter sharing we obtain a simple model which is fast to train and which achieves state-of-the-art or better results on several standard benchmark datasets. Specifically, we use an L1-penalty to regularize the conditional distributions and introduce a procedure for automatic parameter sharing between mixture components. Moreover, we propose a simple distributed representation which permits exact likelihood evaluations since the latent variables are interleaved with the observable variables and can be easily integrated out. Our model achieves excellent generalization performance and scales well to extremely high dimensions.

## 1 INTRODUCTION

We consider the fundamental task of learning a high-dimensional distribution  $\mathbb{P}(\mathbf{x})$  from training examples  $\mathbf{x}^{(n)}$ . The recently most successful approaches are based on autoregressive networks which make use of the decomposition

$$\mathbb{P}(\mathbf{x}) = \prod_{d=1}^D \mathbb{P}(x_d | \mathbf{x}_{1:d-1}).$$

One simple way to model the  $D$  conditional distributions is through logistic or linear regressions (Frey, 1998). Better results can be achieved by using flexible neural networks to model the conditionals (Bengio & Bengio, 2000). Larochelle & Murray (2011) proposed to use the same hidden units for all conditionals. Such a weight sharing among conditional distributions leads to a significant improvement in terms of the generalization performance (Bengio, 2011; Uria et al., 2013). Several variants of this idea have been proposed (Uria et al., 2014; Raiko et al., 2014; Germain et al., 2015). Since the hidden units in these models are deterministic, the associated learning algorithms are relatively simple and exact likelihood evaluations are feasible. Neural networks with stochastic hidden units (Gregor et al., 2014; Bornschein & Bengio, 2015) on the other hand require sophisticated inference procedures and exact likelihood evaluations are then intractable.

In this work we start from sparse logistic and linear models for the conditionals. In contrast to nonsparse nonlinear models, these can be learned from relatively small datasets and have excellent generalization abilities. We then consider large mixtures of such sparse autoregressive networks (SpARN) and study the effect of parameter sharing among mixture components. Since there is no weight sharing between different conditional distributions, our learning procedure can be perfectly parallelized over the data dimensions. We are hence able to train models for extremely high-dimensional data. In our experiments we achieve state-of-the-art or better performance on several standard benchmark problems and obtain convincing results for datasets with several ten thousand dimensions.

## 2 SPARSE AUTOREGRESSIVE NETWORKS

One major source of regularization in neural autoregressive networks is weight sharing among the conditional distributions for different dimensions. This seems necessary in order to avoid overfitting

but makes it hard to parallelize the learning procedure. In addition, early stopping is commonly applied in neural networks to control the model complexity. Other typical regularization techniques are weight decay (Uribe et al., 2013; 2014; Raiko et al., 2014), dropout (Germain et al., 2015) or adaptive weight noise (Gregor et al., 2014). Weight decay means that the L2-norm of the model parameters is penalized.

None of these regularization methods yields sparse parameters. In this work we use an L1-penalty, which induces sparsity. For high-dimensional problems it is very reasonable to assume that only a small number of the variables  $x_1, \dots, x_{d-1}$  are relevant for predicting  $x_d$ . As shown in our experiments, the use of an L1-penalty yields much better results than an L2-penalty. We also tried elastic nets (Zou & Hastie, 2005), which use an L1- as well as an L2-penalty, but the performance was typically worse than with just an L1-penalty. Since the effect of regularization depends on the scale of the variables, we encode binary data as  $\pm 1$  in order not to create a systematic bias. Similarly, for real-valued data we normalize the variables to have mean zero and variance one. The conditional distributions in our autoregressive network are modeled through L1-penalized logistic or linear regressions which are fitted separately for each dimension via coordinate descent.

Concretely, for binary data  $\mathbf{x} \in \{-1, 1\}^D$  we minimize

$$\sum_{n=1}^N \log(1 + \exp(-x_d^{(n)}[\alpha_0^{(d)} + (\boldsymbol{\alpha}_{1:d-1}^{(d)})^T \mathbf{x}_{1:d-1}^{(n)}])) + \lambda_0 |\alpha_0^{(d)}| + \lambda \|\boldsymbol{\alpha}_{1:d-1}^{(d)}\|_1$$

where  $\lambda_0, \lambda > 0$ . The intercept  $\alpha_0^{(d)}$  has a special meaning and is hence regularized differently than the dependency weights  $\alpha_i^{(d)}, i = 1 \dots, d-1$ . Shrinkage for the intercept is needed to avoid potentially degenerated probabilities (in case the empirical variance of  $x_d$  is zero) but also improves the generalization performance. The penalty strength  $\lambda_0$  can be much smaller than  $\lambda$ . The ratio  $\lambda/\lambda_0$  is known as the intercept scaling factor. We do not expect the intercepts to be mostly zero, so a different type of penalty could be used. But for simplicity we penalize the absolute value.

For continuous data  $\mathbf{x} \in \mathbb{R}^D$  we minimize

$$\frac{1}{2} \sum_{n=1}^N (x_d^{(n)} - (\boldsymbol{\alpha}_{1:d-1}^{(d)})^T \mathbf{x}_{1:d-1}^{(n)})^2 + \lambda \|\boldsymbol{\alpha}_{1:d-1}^{(d)}\|_1$$

where  $\lambda > 0$ . The intercept is not needed here since all variables are centered in advance. The parameters  $\boldsymbol{\alpha}_{1:d-1}^{(d)}$  specify the conditional mean  $\mu^{(d)}(\mathbf{x}_{1:d-1})$  of  $x_d$  given  $\mathbf{x}_{1:d-1}$ . The conditional standard deviation  $\sigma^{(d)}$  is assumed to be fixed (i.e., it does not depend on  $\mathbf{x}_{1:d-1}$ ) and is estimated from the residuals  $x_d^{(n)} - \mu^{(d)}(\mathbf{x}_{1:d-1})$ . If each conditional distribution  $x_d|\mathbf{x}_{1:d-1}$  is Gaussian with a constant variance then  $\mathbf{x}$  has a multivariate normal distribution. Alternatively, we could model the variance as a, say, quadratic function of the predicted value  $\mu^{(d)}(\mathbf{x}_{1:d-1})$  in which case  $\mathbf{x}$  would no longer be multivariate Gaussian. In any case, our conditional distributions are unimodal. We deal with multimodality by learning mixtures of autoregressive networks. In contrast to that, Bishop (1994); Davies & Moore (2000) model the conditional distributions through mixtures but only learn a single network.

L1-penalized logistic and linear regressions for neighborhood selection have been used before in undirected graphical models (Meinshausen & Bühlmann, 2006; Ravikumar et al., 2010). Sparse linear models for conditional distributions have been used in image analysis by Cressie & Davidson (1998); Domke et al. (2008). However, in these models the relevant parents for each pixel were chosen manually, consisting of a small number of adjacent pixels. This means that no long-range dependencies can be captured. Very restricted neighborhoods can lead to visible artifacts in samples from such models. A fully Bayesian approach for deep sigmoid belief nets with a sparsity-inducing prior has been used in Gan et al. (2015), where the posterior distribution on model parameters is approximated through a variational Bayes procedure. The quantitative performance of this method however is significantly worse than our L1-penalized regression approach (see Section 3).

## 2.1 MIXTURES

For large datasets a sparse autoregressive network is not competitive with more sophisticated networks. Since learning requires only relatively few training examples it is natural to consider mixtures

of the form

$$\mathbb{P}(\mathbf{x}) = \sum_{k=1}^K \mathbb{P}(h = k) \prod_{d=1}^D \mathbb{P}(x_d | \mathbf{x}_{1:d-1}, h)$$

where the conditional distributions are sparse regressions. The latent variable  $h \in \{1, \dots, K\}$  indicates the active mixture component. It is straightforward to learn such a mixture using the EM-algorithm (Dempster et al., 1977). A good initialization can be obtained by first learning a mixture of product (Bernoulli or normal) distributions for the data.

### 2.1.1 WITHOUT PARAMETER SHARING

In a classical mixture model no parameters are shared among the components. The form<sup>1</sup> of the component conditionals in this case is

$$\mathbb{P}(x_d = 1 | \mathbf{x}_{1:d-1}, h) = \sigma(\beta_0^{(h,d)} + (\beta_{1:d-1}^{(h,d)})^T \mathbf{x}_{1:d-1}).$$

Consequently, separate autoregressive networks are trained for different clusters of the data. Without parameter sharing, a typical number of mixture components before overfitting occurs is around one per one thousand training examples.

### 2.1.2 WITH PARAMETER SHARING

A reasonable simplification is to assume that the dependence structure is rather universal, i.e., it does not change much for the different mixture components. With shared dependency weights a much larger mixture can be trained because only separate intercepts have to be learned. The form of the component conditionals with parameter sharing is

$$\mathbb{P}(x_d = 1 | \mathbf{x}_{1:d-1}, h) = \sigma(\beta_0^{(h,d)} + (\boldsymbol{\alpha}_{1:d-1}^{(d)})^T \mathbf{x}_{1:d-1}).$$

The same sharing of dependency weights is used in autoregressive sigmoid belief networks (Gregor et al., 2014), which are large implicit mixtures of (dense) autoregressive networks. Note that this kind of parameter sharing across mixture components is very different from the mentioned weight sharing in neural autoregressive networks (e.g., Larochelle & Murray, 2011) where weights are shared across conditionals for different dimensions. With shared dependency weights, a typical number of mixture components before overfitting occurs is around one per one hundred training examples.

### 2.1.3 AUTOMATIC PARAMETER SHARING

Rather than manually deciding which parameters to share we can let the amount of sharing be part of the learning process. Specifically, we can use a global bias term and global dependency weights, and let the component parameters describe deviations from these global parameters. The form of the component conditionals in this case is

$$\mathbb{P}(x_d = 1 | \mathbf{x}_{1:d-1}, h) = \sigma(\alpha_0^{(d)} + \beta_0^{(h,d)} + (\boldsymbol{\alpha}_{1:d-1}^{(d)} + \boldsymbol{\beta}_{1:d-1}^{(h,d)})^T \mathbf{x}_{1:d-1}).$$

We want the component parameters only to be nonzero if there is substantial gain from untying the corresponding weights. Consequently, we use the penalty

$$\lambda_0 |\alpha_0^{(d)}| + \lambda_0 \sum_{h=1}^K |\beta_0^{(h,d)}| + \lambda \|\boldsymbol{\alpha}_{1:d-1}^{(d)}\|_1 + \lambda \sum_{h=1}^K \|\boldsymbol{\beta}_{1:d-1}^{(h,d)}\|_1.$$

A similar penalty for deviations from a shared parameter has been used for multi-task learning with SVMs (Evgeniou & Pontil, 2004). The optimal number of mixture components with automatic sharing is between the optimal numbers of components for mixtures with and without parameter sharing, respectively. In our experiments, mixtures with automatic parameter sharing always performed better than the mixtures with or without parameter sharing.

<sup>1</sup>We use here the notation for binary data, the corresponding expressions for continuous data are evident.

## 2.2 SEQUENCE OF MIXTURES

In a mixture model only a single component is active at a time. In contrast to that, a distributed representation (Bengio et al., 2013) allows several experts to cooperate in order to explain the observations. Each expert is a model for the data vector  $\mathbf{x}$ . Using a specified composition rule (Goessling & Amit, 2015) the active experts are combined to create the composed model for  $\mathbf{x}$ . The traditional approach is to use a top-down model in which latent variables are generated first and the distribution of the visible variables is modeled conditioned on all hidden variables. Such an organization of the variables makes inference and learning difficult. Moreover, exact likelihood evaluations are usually intractable because the required computations involve a sum over all configurations of the latent variables.

We propose a simple distributed representation for which inference is trivial and exact likelihood evaluations are tractable. The basic idea is to divide the data dimensions into chunks and to learn local models for each of these subsets of variables. Specifically, we partition the  $D$  dimensions into  $L$  disjoint intervals  $I_\ell := [d_{\ell-1} + 1 : d_\ell]$  where

$$0 = d_0 < d_1 < \dots < d_L = D.$$

We then use mixtures of sparse autoregressive networks for each of the data vectors  $\mathbf{x}_{I_\ell}$ ,  $\ell = 1, \dots, L$ , utilizing the data  $\mathbf{x}_{1:d_{\ell-1}}$  from previous dimensions as additional predictors, i.e.,

$$\mathbb{P}(\mathbf{x}_{I_\ell} | \mathbf{x}_{1:d_{\ell-1}}, h_\ell) = \prod_{d=d_{\ell-1}+1}^{d_\ell} \mathbb{P}(x_d | \mathbf{x}_{1:d-1}, h_\ell).$$

Associated with each mixture is a latent variable  $h_\ell \in \{1, \dots, K_\ell\}$  which specifies the component that is active for interval  $I_\ell$ . By choosing different configurations of mixture components global variation is achieved while the autoregressive networks themselves capture local variation of the data. For each choice of latent variables, the combined model is an autoregressive network for  $\mathbf{x}$ .

Partition-based representations are simpler and more interpretable than other distributed representations because the area of responsibility is explicitly defined for the local models. Moreover, since the local models describe disjoint sets of variables no composition rule is required (for each observable variable exactly one local model is employed). For image modeling the main drawbacks of existing partition-based approaches (Pal et al., 2002; Aghajanian & Prince, 2008) are discontinuities and misalignments along the partition boundaries. Since in our approach each local model is an autoregressive network that uses the data from all previous dimensions as predictors (not just the data within its own interval  $I_\ell$ ) these problems are largely eliminated, see the experimental results in Section 3.

What remains to be specified is the distribution on the latent variables. In top-down approaches, for example deep autoregressive networks (Gregor et al., 2014), the joint distribution on latent and observable variables is factored as  $\mathbb{P}(\mathbf{h}, \mathbf{x}) = \mathbb{P}(\mathbf{h})\mathbb{P}(\mathbf{x}|\mathbf{h})$ . This makes it easy to evaluate  $\mathbb{P}(\mathbf{h})$  but hard to evaluate  $\mathbb{P}(\mathbf{x})$ . Rather than first generating all latent variables we propose to interleave them with the observable variables. Formally, the joint distribution is factored as

$$\mathbb{P}(\mathbf{h}, \mathbf{x}) = \prod_{\ell=1}^L \mathbb{P}(h_\ell | \mathbf{x}_{1:d_{\ell-1}}) \mathbb{P}(\mathbf{x}_{I_\ell} | \mathbf{x}_{1:d_{\ell-1}}, h_\ell).$$

where  $\mathbb{P}(h_\ell | \mathbf{x}_{1:d_{\ell-1}})$  are multiclass logistic regressions. This means in particular that given  $\mathbf{x}_{1:d_{\ell-1}}$  the distribution for  $h_\ell$  does not depend on  $\mathbf{h}_{1:\ell-1}$ . Inference is thus trivial since the posterior distribution  $\mathbb{P}(\mathbf{h}|\mathbf{x})$  factorizes over the latent variables. In contrast to that, hidden Markov models (Rabiner, 1989) define the distribution of  $h_\ell$  in terms of  $h_{\ell-1}$ . Our factorization also makes it easy to integrate out the latent variables. Indeed, it follows that

$$\mathbb{P}(\mathbf{x}) = \prod_{\ell=1}^L \sum_{h_\ell=1}^{K_\ell} \mathbb{P}(h_\ell | \mathbf{x}_{1:d_{\ell-1}}) \mathbb{P}(\mathbf{x}_{I_\ell} | \mathbf{x}_{1:d_{\ell-1}}, h_\ell).$$

Training the model hence reduces to learning  $L$  separate mixtures of sparse autoregressive networks because the full likelihood  $P(\mathbf{x})$  factorizes over  $\ell$ . Consequently, the model can be trained by

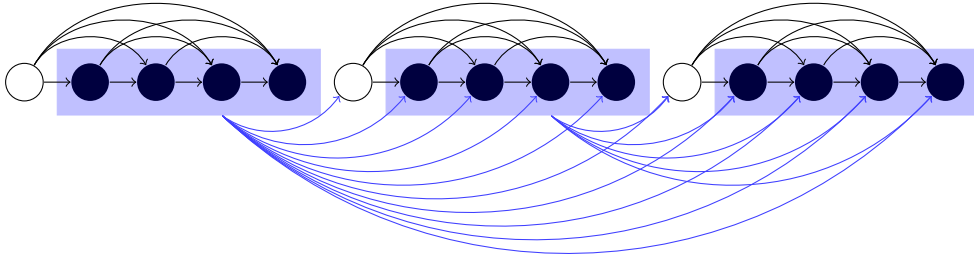


Figure 1: Graphical representation of the sequence-of-mixtures model. Solid nodes represent observable variables and empty nodes represent latent variables. Blue edges denote dependence on a block of variables.

learning  $L$  separate mixtures of sparse autoregressive networks. A graphical representation of this sequence-of-mixtures model is shown in Figure 1. The proposed model family is a subfamily of sum-product networks (Poon & Domingos, 2011) which is particularly easy to train and conceptually simple.

### 3 EXPERIMENTS

We quantitatively evaluated our sparse autoregressive networks (SpARN) on various high-dimensional datasets. For all experiments we used an intercept scaling factor (see Section 2) of 10, meaning the penalty on the intercept is a factor 10 smaller than for the dependency weights. The exact choice of the factor is not crucial, other values between 10 and 100 lead to very similar results. We used likelihood evaluations on the validation set to choose the number of mixture components and to select the penalty strength  $\lambda$ . The same penalty strength was applied for every dimension. We also performed a qualitative evaluation on two very high-dimensional datasets.

#### 3.1 CALTECH-101 SILHOUETTES

The first dataset we considered are the Caltech-101 silhouettes (Marlin et al., 2010). There are 4,100 training samples, 2,264 validation samples and 2,307 test samples of dimension  $28 \times 28 = 784$ . Each data point is a binary mask for an object from one of 101 categories. Since the similarity between samples is of semantic nature a large degree of abstraction is required to generalize well to unseen examples. Previous results for this dataset are shown in Table 1. The current state of the art is a logistic autoregressive network, also called a fully visible sigmoid belief net (FVSBN), which was learned through a variational Bayes procedure (Gan et al., 2015). Our network based on L1-penalized logistic regressions and trained through coordinate descent achieves an average test log-likelihood which is 4.5 nats higher. An autoregressive network with L2-penalized logistic regressions performs significantly worse. The sparsity induced by the L1-penalty is thus crucial. We also trained mixtures of sparse autoregressive networks in an unsupervised manner (i.e., without using any category labels). The mixture with shared dependency weights (tied) improves over the single network by more than 2 nats. The optimal number of mixture components is 100, which makes sense because that is about the number of different object categories in the dataset. When trained without intercept scaling the mixture essentially reduced to a single sparse network. Hence, it was important to have a weaker penalty on the intercepts. The mixture with automatic parameter sharing (auto) further improves the test performance by about 1 nat. The mixture without parameter sharing (untied) was not better than the single network.

#### 3.2 BINARIZED MNIST DIGITS

The next dataset we considered are the MNIST digits (Salakhutdinov & Murray, 2008) which were binarized through random sampling based on the original intensity value. This is a relatively large dataset consisting of 50,000 training samples, 10,000 validation samples and 10,000 test samples again of dimension  $28 \times 28 = 784$ . Previous results for this dataset are shown in Table 1. A fully visible sigmoid belief net trained by stochastic gradient descent and regularized through early

Table 1: Average log-likelihoods (in nats) per test example using different models. The best result for each dataset is shown in bold. Baseline results are taken from [1]Raiko et al. (2014), [2]Bornschein & Bengio (2015), [3]Gan et al. (2015), [4]Larochelle & Murray (2011), [5]Germain et al. (2015) and [6]Gregor et al. (2014).

Caltech-101 silhouettes		Binarized MNIST digits			
RBM [1]	$\approx -107.78$	FVSBN (VB) [3]	$\approx -100.76$	SpARN (1 comp)	-97.34
NADE-5 [1]	-107.28	FVSBN (SGD) [4]	-97.45	SpARN (20 comp, untied)	-89.43
RWS-NADE [2]	$\approx -104.30$	NADE [4]	-88.86	SpARN (50 comp, auto)	-87.95
FVSBN (VB) [3]	$\approx -96.40$	MADE [5]	-86.64	SpARN (500 comp, tied)	-91.32
ARN (1 comp, L2-penalty)	-95.95	RBM (CD-25) [4]	$\approx -86.34$	SpARN (4 $\times$ 50 comp, auto)	-87.40
SpARN (1 comp)	-91.80	DARN [6]	$\approx -84.13$	SpARN (4 $\times$ 500 comp, tied)	-88.63
SpARN (100 comp, auto)	<b>-88.48</b>				
SpARN (100 comp, tied)	-89.59				

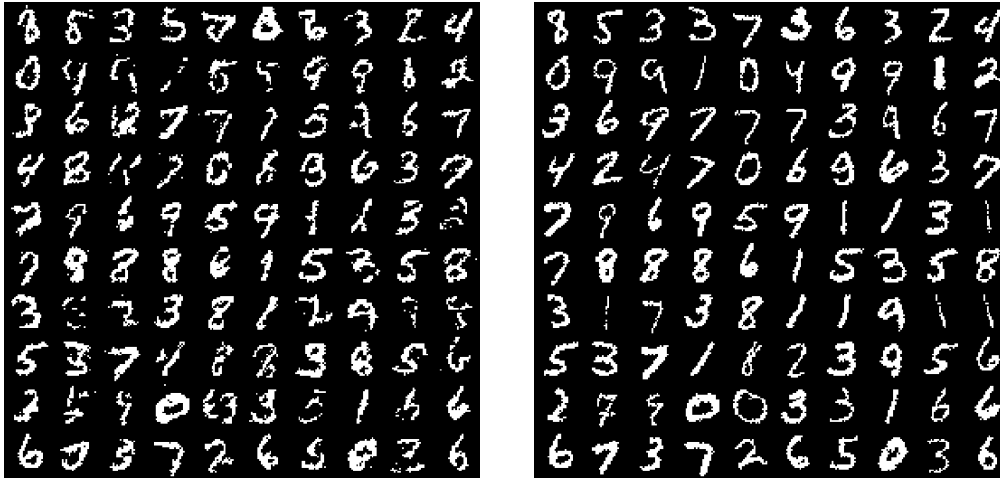


Figure 2: **Left:** Samples from the sequence-of-mixtures model trained on MNIST digits. **Right:** Closest training examples.

stopping achieves about the same performance as a sparse logistic autoregressive network. This is because the dataset is rather large and hence the need for regularization is small. For mixtures without parameter sharing the optimal number of components is 20. Using automatic parameter sharing it is 50 and with shared dependency weights it is 500. All mixtures substantially improve over the single network. For this dataset, the mixture without sharing performs better than the mixture with shared parameters. We believe that this is the case because compared to the Caltech-101 silhouettes the distribution of MNIST digits has fewer modes and more training samples are available to learn from. We also trained a sequence-of-mixtures model. For that the image grid was divided into four quadrants of size  $14 \times 14$  pixels and a mixture model was learned for each of the quadrants using the same number of components as before. With parameter sharing this model improves the performance of the corresponding mixture by 2.5 nats. Note that the number of parameters for the sequence-of-mixtures model is almost the same as for the mixture. The only additional parameters come from the multiclass logistic regressions for the hidden variables. When automatic parameter sharing is used the gain through a sequence of mixtures is smaller. Without parameter sharing there is no gain from the sequence of mixtures. Samples from the sequence-of-mixtures model with automatic parameter sharing as well as nearest training examples are shown in Figure 2. This shows that the model puts the probability mass in the right region of the data space and that it does not simply memorize the training examples.

### 3.3 BINARY UCI DATASETS

We performed additional quantitative evaluations on eight standard benchmark datasets from the UCI Machine Learning Repository using the same splits into training, validation and test sets as in previous work. The datasets have various sample sizes and dimensions. The ratio of training sam-

Table 2: Average log-likelihoods (in nats) per test example for various models and datasets. The best result for each dataset is shown in bold. Baseline results are taken from Germain et al. (2015); Bornschein & Bengio (2015). Dataset sizes, standard errors and used numbers of mixture components are shown in italic.

	<b>Adult</b>	<b>Connect4</b>	<b>DNA</b>	<b>Mushrooms</b>	<b>NIPS-0-12</b>	<b>OCR-letters</b>	<b>RCV1</b>	<b>Web</b>
<i>train</i>	<i>5,000</i>	<i>16,000</i>	<i>1,400</i>	<i>2,000</i>	<i>400</i>	<i>32,152</i>	<i>40,000</i>	<i>14,000</i>
<i>valid</i>	<i>1,414</i>	<i>4,000</i>	<i>600</i>	<i>500</i>	<i>100</i>	<i>10,000</i>	<i>10,000</i>	<i>3,188</i>
<i>test</i>	<i>26,147</i>	<i>47,557</i>	<i>1,186</i>	<i>5,624</i>	<i>1,240</i>	<i>10,000</i>	<i>150,000</i>	<i>32,561</i>
<i>dim</i>	<i>123</i>	<i>126</i>	<i>180</i>	<i>112</i>	<i>500</i>	<i>128</i>	<i>150</i>	<i>300</i>
Ber. Mix.	-20.44	-23.41	-98.19	-14.46	-290.02	-40.56	-47.59	30.16
FVSBN (SGD)	-13.17	-12.39	-83.64	-10.27	-276.88	-39.30	-49.84	-29.35
NADE	-13.19	-11.99	-84.81	-9.81	-273.08	-27.22	-46.66	-28.39
DARN	-13.19	-11.91	-81.04	-9.55	-274.68	$\approx$ -28.17	$\approx$ -46.10	$\approx$ -28.83
MADE	-13.12	-11.90	-79.66	-9.68	-277.28	-28.34	-46.74	-28.25
RWS-NADE	$\approx$ -13.16	$\approx$ -11.68	$\approx$ -84.26	$\approx$ -9.71	$\approx$ -271.11	$\approx$ -26.43	$\approx$ -46.09	$\approx$ -27.92
SpARN (untied)	<b>-13.04</b>	-12.04	-79.32	-9.58	-271.13	-28.48	-45.55	-27.98
SpARN (auto)	<b>-13.04</b>	-11.98	<b>-79.05</b>	<b>-9.38</b>	<b>-270.29</b>	-27.96	<b>-45.11</b>	<b>-27.50</b>
SpARN (tied)	<b>-13.04</b>	-12.14	-79.26	-9.39	-270.53	-31.53	-45.50	-28.12
<i>std. error</i>	<i>0.02</i>	<i>0.01</i>	<i>0.21</i>	<i>0.01</i>	<i>0.52</i>	<i>0.11</i>	<i>0.06</i>	<i>0.10</i>
<i>comp (untied)</i>	<i>1</i>	<i>10</i>	<i>1</i>	<i>2</i>	<i>1</i>	<i>100</i>	<i>200</i>	<i>100</i>
<i>comp (auto)</i>	<i>1</i>	<i>10</i>	<i>3</i>	<i>10</i>	<i>20</i>	<i>200</i>	<i>1,000</i>	<i>200</i>
<i>comp (tied)</i>	<i>1</i>	<i>10</i>	<i>3</i>	<i>20</i>	<i>20</i>	<i>5,000</i>	<i>5,000</i>	<i>200</i>

ples to dimensionality varies between 1 and 300. Table 2 reports the obtained test log-likelihoods for mixtures of sparse autoregressive networks together with previous results. The table also shows the optimal numbers of mixture components for the different kinds of parameter sharing and the standard error of the test log-likelihood (when using automatic parameter sharing). On smaller datasets mixtures with parameter sharing tend to perform a bit better than mixtures without parameter sharing, and vice versa on larger datasets. The performance of our mixtures of sparse autoregressive networks with automatic parameter sharing is better than the best previously reported result on six of the eight datasets (all improvements except for the NIPS-0-12 dataset are statistically significant). For the other two datasets the performance is close to the state of the art.

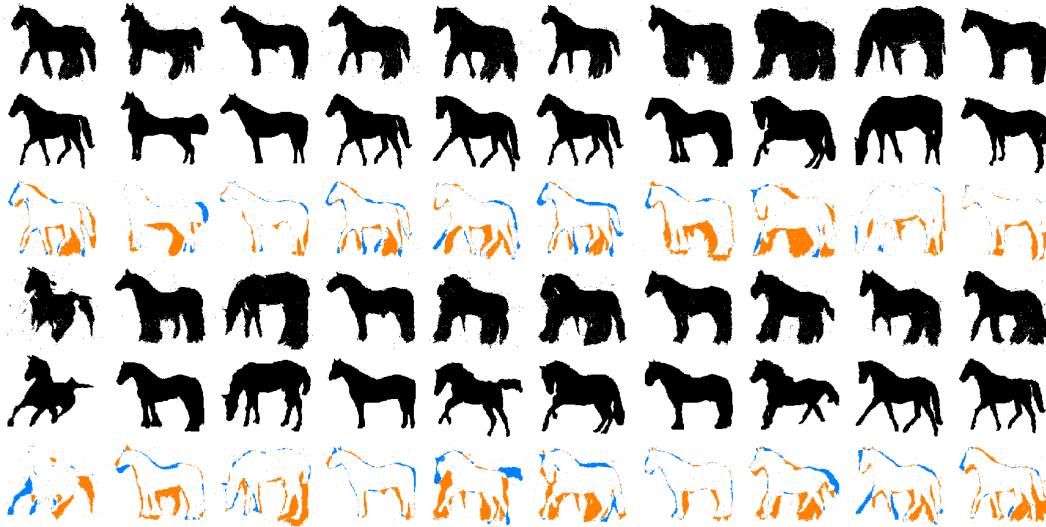


Figure 3: **1st & 4th row:** Samples from the sequence-of-mixtures model trained on Weizmann horses. **2nd & 5th row:** Closest training examples. **3rd & 6th row:** Symmetric differences between the synthetic sample and the closest training example (pixels in blue are only ‘on’ for the training example and pixels in orange are only ‘on’ for the synthetic sample).

### 3.4 WEIZMANN HORSES

The following experiment shows the ability of our model to cope with very high-dimensional data. The Weizmann horse dataset (Borenstein & Ullman, 2008) consists of 328 binary images of size  $200 \times 240$ . We divided the image grid into four quadrants of size  $100 \times 120$  and learned a mixture of sparse autoregressive networks for each quadrant. We decided to use five components each for the top two quadrants (i.e., horse head and back) and ten components each for the bottom two quadrants (i.e., horse legs) because there is more variability in the lower half of the image. Samples from the sequence-of-mixtures model are presented in Figure 3 together with the closest training examples. Our model creates overall realistic samples which differ from the training examples. In particular, no discontinuities along the borders of the quadrants are visible. We emphasize that our model was trained using the full resolution of the images. Each data point is a 48,000 dimensional binary vector. The same dataset was used in Eslami et al. (2014) where a Boltzmann machine was learned from the data. In their experiments the images were downsampled to  $32 \times 32$  pixels, which is a factor 50 smaller than in our case. This was probably necessary in order to speed up computations and to avoid overfitting. Refer to Figure 5(d) in that paper to compare the quality of the model samples.

### 3.5 CALTECH FACES

A very high-dimensional real-valued dataset that we considered are the Caltech Faces (Fergus et al., 2003) consisting of 450 high-resolution color images. We downsampled the images to  $200 \times 150 \times 3$ , which are 90,000 dimensions. Samples from a sparse linear autoregressive network with constant conditional variance are presented in Figure 4. The samples are diverse and show substantial abstraction from the training examples. We also learned a network in which the conditional variance was modeled as a quadratic function of the predicted value. This improved the likelihood only slightly and the corresponding samples were visually indistinguishable from the samples of the network with constant variance. Since the variability in this dataset is mainly local, a mixture of sparse autoregressive networks did not improve over the single network in terms of likelihoods.



Figure 4: **First five columns:** Samples from a sparse linear autoregressive network trained on Caltech faces. **Last column:** Closest training examples for the adjacent sample.



## 4 CONCLUSION

We pointed out the importance of sparsity in autoregressive networks. Using simple mixtures of sparse autoregressive networks we achieved results which are competitive with much more sophisticated models. We also introduced a distributed representation for autoregressive networks based on a sequence of mixture models. No weight sharing across dimensions was needed for our models which allowed us to perfectly parallelize the learning procedure.

## REFERENCES

- Jania Aghajanian and Simon JD Prince. Mosaicfaces: a discrete representation for face recognition. In *IEEE Workshop on Applications of Computer Vision*, pp. 1–8, 2008.
- Samy Bengio and Yoshua Bengio. Taking on the curse of dimensionality in joint distributions using neural networks. *IEEE Transactions on Neural Networks*, 11(3):550–557, 2000.
- Yoshua Bengio. Discussion of the neural autoregressive distribution estimator. In *International Conference on Artificial Intelligence and Statistics*, pp. 38–39, 2011.
- Yoshua Bengio, Aaron Courville, and Pierre Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.
- Christopher M Bishop. Mixture density networks. Technical Report NCRG/94/004, Aston University, Birmingham, 1994.
- Eran Borenstein and Shimon Ullman. Combined top-down/bottom-up segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(12):2109–2125, 2008.
- Jörg Bornschein and Yoshua Bengio. Reweighted wake-sleep. In *International Conference on Learning Representations*, 2015.
- Noel Cressie and Jennifer L Davidson. Image analysis with partially ordered markov models. *Computational Statistics & Data Analysis*, 29(1):1–26, 1998.
- Scott Davies and Andrew Moore. Mix-nets: Factored mixtures of gaussians in bayesian networks with mixed continuous and discrete variables. In *Conference on Uncertainty in Artificial Intelligence*, pp. 168–175, 2000.
- Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (methodological)*, pp. 1–38, 1977.
- Justin Domke, Alap Karapurkar, and Yiannis Aloimonos. Who killed the directed model? In *Conference on Computer Vision and Pattern Recognition*, pp. 1–8, 2008.
- SM Ali Eslami, Nicolas Heess, Christopher KI Williams, and John Winn. The shape boltzmann machine: a strong model of object shape. *International Journal of Computer Vision*, 107(2): 155–176, 2014.
- Theodoros Evgeniou and Massimiliano Pontil. Regularized multi-task learning. In *International conference on knowledge discovery and data mining*, pp. 109–117, 2004.
- Robert Fergus, Pietro Perona, and Andrew Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Conference on Computer Vision and Pattern Recognition*, volume 2, pp. 264–271, 2003.
- Brendan J Frey. *Graphical models for machine learning and digital communication*. MIT press, 1998.
- Zhe Gan, Ricardo Henao, David Carlson, and Lawrence Carin. Learning deep sigmoid belief networks with data augmentation. In *International Conference on Artificial Intelligence and Statistics*, pp. 268–276, 2015.

- Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, pp. 881–889, 2015.
- Marc Goessling and Yali Amit. Compact part-based image representations. In *International Conference on Learning Representations (Workshop)*, 2015.
- Karol Gregor, Ivo Danihelka, Andriy Mnih, Charles Blundell, and Daan Wierstra. Deep autoregressive networks. In *International Conference on Machine Learning*, pp. 1242–1250, 2014.
- Hugo Larochelle and Iain Murray. The neural autoregressive distribution estimator. In *International Conference on Artificial Intelligence and Statistics*, pp. 29–37, 2011.
- Benjamin M Marlin, Kevin Swersky, Bo Chen, and Nando D Freitas. Inductive principles for restricted boltzmann machine learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 509–516, 2010.
- Nicolai Meinshausen and Peter Bühlmann. High-dimensional graphs and variable selection with the lasso. *The Annals of Statistics*, pp. 1436–1462, 2006.
- Chris Pal, Brendan J Frey, and Nebojsa Jojic. Learning montages of transformed latent images as representations of objects that change in appearance. In *Computer Vision – ECCV*, pp. 715–731, 2002.
- Hoifung Poon and Pedro Domingos. Sum-product networks: A new deep architecture. In *International Conference on Computer Vision (Workshops)*, pp. 689–690, 2011.
- Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.
- Tapani Raiko, Yao Li, Kyunghyun Cho, and Yoshua Bengio. Iterative neural autoregressive distribution estimator nade-k. In *Advances in Neural Information Processing Systems*, pp. 325–333, 2014.
- Pradeep Ravikumar, Martin J Wainwright, and John D Lafferty. High-dimensional ising model selection using  $\ell_1$ -regularized logistic regression. *The Annals of Statistics*, 38(3):1287–1319, 2010.
- Ruslan Salakhutdinov and Iain Murray. On the quantitative analysis of deep belief networks. In *International Conference on Machine learning*, pp. 872–879, 2008.
- Benigno Uria, Iain Murray, and Hugo Larochelle. Rnade: The real-valued neural autoregressive density-estimator. In *Advances in Neural Information Processing Systems*, pp. 2175–2183, 2013.
- Benigno Uria, Iain Murray, and Hugo Larochelle. A deep and tractable density estimator. In *International Conference on Machine Learning*, pp. 467–475, 2014.
- Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.